[0101] Referring now to **FIG. 2**, the object inspector **30** is shown in more detail. The object inspector **30** provides two columns, an attribute name column **100** and a property column **110**. The object being inspected in **FIG. 2** has properties **112**, **114**, **116**, **118**, **120**, **122**, **124** and **126**. Particularly, properties **114**, **116**, **118**, **120**, **124** and **126** are static constants. However, properties **112** and **122** are dynamic and are evaluated at run time. Due to the dynamic properties **112** and **122**, to create a data driven application, the user only needs to enter the dynamic properties and code may be automatically generated. Thus, the data binding of the dynamic properties **112** and **122** is seamlessly integrated into the property sheet **30**.

[0102] The Object Inspector window **30** is a dockable control bar which displays an object's properties and events. The Object Inspector may be resized horizontally when docked and both vertically and horizontally when floating. The window may dock to the left or right of the world workspace when its location overlaps the docking site while being dragged. The Object Inspector **30** is displayed when a scene editor or data element editor window is active in design mode and displays a drop-down combobox containing a list of all objects within scope of the editor. A "dot" notation is used to denote compound objects. For example, "ScatterChart1.DataElement1" denotes that DataElement1 is an object owned by ScatterChart1.

[0103] Beneath the object selection combobox is a tabbed sheet displaying either the properties or the events for an object. When the properties tab is selected, a 2-column entry form is displayed for setting the values of the selected object's properties. The left column is a read-only column that displays the name of the property, while the right column is a read/write column that displays the property's value. Values entered in the right column may be constants or may be calculated values containing functions of parameters or column names from a data source. When a property value is of an enumerated type (a value that can only be set to a finite number of values), a drop-down combobox may be displayed listing the legal values for the property. Enumerated types include but are not limited to the following:

[0104] Font name (e.g., "Arial", "Courier")

[0105] Color (e.g., Red, Blue)

[0106] Text alignment (e.g., Left, Right)

[0107] Text anchor points (e.g., LowerLeft, UpperRight)

[0108] Boolean values (e.g., True or False)

[0109] Line style (e.g., Solid, Dash)

[0110] Fill pattern (e.g., Solid, Horizontal)

[0111] When a property is a date or time, a calendar control may be displayed beneath the property value when the field is active.

[0112] Form and data element objects provide control over the values of query parameters through a QueryParameters object listed in the Object Inspector. The QueryParameters object is a child object of the form or data element and contains a property for each query parameter. If the query is not parameterized, the QueryParameters object is not displayed in the Object Inspector. The values contained by the object are used to set the query's properties at runtime before the query is executed.

[0113] **FIG. 3** shows a property object model of an abstract base class VcPropertyBag **200**. VcPropertyBag **200** acts as a container for a set of properties that are specific to one or more derived classes. VcPropertyBag **200** provides an interface for accessing and manipulating properties and their values. It also provides a pure virtual method for obtaining type information on the derived object's set of properties.

[0114] Derived from VcPropertyBag **200** are an abstract base class VcScene **202** and an abstract base class VcDrawingNode **204**. VcScene **202** is the class for a scene and inherits properties from VcPropertyBag **200**. VcDrawingNode **204** is an abstract base class for all graphical objects displayed in a scene and also inherits properties from VcPropertyBag **200**. VcDrawingNode **204** provides a set of pure virtual methods for manipulating properties in support of movement, scaling, and other standard editing operations. Derived classes are responsible for mapping the editing operations to the appropriate properties.

[0115] VcPropertyBag **200** also has a set of properties m_properties of class VcProperty **210**, which contains all information about a specific object property, including name and enumerated type information, m_expectedType. m_expectedType describes the data type of the resulting property value (e.g., boolean, numeric, string, etc.), stores the path to access the object m_path, and manages a design-time value m_designvalue and a run-time value m_runtimevalue for the property. The design-time always evaluates to a constant so that the container object may be displayed in isolation (without links to data or parameters). When the run-time value is also constant, the two values are set to the same value.

[0116] VcProperty **210** has an m_name property which belongs to CString **212**, a standard string class to provide storage for the property's name. The VcProperty object **210** also has a m_path property of a class called VcPropertyPath **214**, which stores an identifier for a property that is unique within the VcPropertyBag's set of properties, and which provides a pointer, m next, to a linked list of VcPropertyPath to support aggregated property bags.

[0117] VcProperty **210** also has a property m_design value which belongs to a class called VcPropertyValue **216**. VcPropertyValue **216** stores a function representing the design-time value for the property (VcFunction **220**) and stores the expression entered by the user so that case-sensitivity, spacing, and other particulars may be displayed back to the user as entered (CString **218**).

[0118] VcFunction **220** is an abstract base class for a parsed expression element and the function is the root element for an expression tree that may be evaluated by supplying a context object. CString **218** is a standard string class which provides storage for the expression entered by the user.

[0119] Finally, VcProperty object **210** also has a m_runtime value property which belongs to an abstract base class VcPropertyValue **222**. VcPropertyValue **222** stores a function representing the run-time value for the property and